IN THE CLAIMS

The text of all claims, along with their current status, is set forth below:

1. (Previously Presented) A method of downloading data sets from among a plurality of host computers, comprising the steps of:

(a) storing representations of data set addresses in a set of data structures, including a buffer and a first disk file, wherein the representations of data set addresses stored in the first disk file are ordered;

(b) downloading at least one data set that includes addresses of one or more referred data sets;

(c) identifying the addresses of the one or more referred data sets;

(d) for each identified address:

(d1) generating a representation of the identified address;

(d2) determining whether the representation is stored in the buffer without determining whether the representation is stored in the first disk file, and when this determination is negative, storing the representation in the buffer; and

(e) when the buffer reaches a predefined full condition:

(e1) ordering the contents of the buffer according to the representations; and

(e2) performing an ordered merge of the contents of the buffer into the contents of the first disk file wherein the ordered merge comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

2. (Original) The method of claim 1, further comprising:

in step (d2), when the determination is negative, storing the identified address in the buffer.

3. (Original) The method of claim 1, further comprising:

in step (d2), when the determination is negative, storing the identified address in a second disk file;

in step (d2), additionally storing with each representation in the buffer a pointer to the corresponding address stored in the second disk file; and

in step (e1), while ordering the contents of the buffer, keeping with each representation in the buffer its pointer to the corresponding address in the second disk file.

4. (Original) The method of claim 3 wherein

step (e2) includes: for each representation in the buffer storing an associated flag, setting the flag to a first value when the representation is equal to a representation previously stored in the first disk file, and setting the flag to a second value, distinct from the first value, when the representation is not equal to any representation previously stored in the first disk file; and

step (e) includes: (e3) for each representation whose flag is set to the second value, scheduling the corresponding data set for downloading.

5. (Original) The method of claim 1 wherein:

step (a), storing representations of data set addresses, includes the step of storing

representations of data set addresses in a sparse disk file which is divided into portions, each portion having a starting address and contents comprising an ordered list of representations of data addresses; and

step (e2), merging the contents of the buffer with the ordered contents of the sparse disk file, includes:

for each of a plurality of the representations stored in the buffer:

(e2-1) determining a starting address for a corresponding portion of the sparse disk file; and

(e2-2) performing an ordered merge of a subset of the buffer, starting at the representation for which the starting address was obtained, into the contents of the corresponding portion.

6. (Original) The method of claim 1 wherein:

step (a), storing representations of data set addresses, includes the step of storing representations of data set addresses in a sparse disk file having empty entries interspersed among entries storing said representations; and

step (e2), merging the contents of the buffer with the ordered contents of the sparse disk file, includes:

for each respective representation stored in the buffer:

(e2-1) determining a starting address for a corresponding portion of the sparse disk file; and

(e2-2) sequentially scanning the disk file, starting at the representation for which the starting address was obtained, until the first of (A) a representation matching the respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

7. (Original) The method of claim 1 wherein, in step (dl), the representation comprises a checksum of at least a portion of the identified address.

8. (Original) The method of claim 1 wherein step (d2) further comprises:

(d2-l) determining whether the representation is stored in a cache before determining whether the representation is stored in the buffer;

(d2-2) when the representation is not stored in the cache, the cache has not reached a predefined full condition, and other predefined criteria are met, adding the representation to the cache; and

(d2-3) when the representation is not stored in the cache, the cache has reached said predefined full condition, and said other predefined criteria are met, evicting a stored representation from the cache in accordance with an eviction policy and adding the representation to the cache.

9. (Original) The method of claim 1 wherein step (e2) further comprises:

when a representation in the first buffer is not found in the first disk file during merging, scheduling the corresponding data set for downloading.

10. (Original) The method of claim 8 wherein step (e2) further comprises:

when a representation in the buffer is not found in the first disk file during merging, scheduling the corresponding data set for downloading.

11. (Original) The method of claim 8 wherein:

step (a), storing representations of data set addresses, includes the step of storing representations of data set addresses in a sparse disk file which is divided into portions, each portion having a starting address and contents comprising an ordered list of representations of data addresses; and

step (e2), performing an ordered merge of the contents of the buffer into the contents of the sparse disk file, includes:

for each of a plurality of the representations stored in the buffer:

(e2-1) obtaining a starting address for a corresponding portion of the sparse disk file; and

(e2-2) performing an ordered merge of a subset of the buffer, starting at the representation for which the starting address was obtained, into the contents of the corresponding portion.

12. (Original) The method of claim 8 wherein:

step (a), storing representations of data set addresses, includes the step of storing representations of data set addresses in a sparse disk file having empty entries interspersed among entries storing said representations; and

step (e2), merging the contents of the buffer with the ordered contents of the sparse disk file, includes:

for each respective representation stored in the buffer:

(e2-1) determining a starting address for a corresponding portion of the sparse disk file; and

(e2-2) sequentially scanning the disk file, starting at the representation for which the starting address was obtained, until the first of (A) a representation matching the

respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

13. (Previously Presented) A method of downloading data sets from among a plurality of host computers, comprising the steps of:

(a) storing representations of data set addresses in a set of data structures, including a first buffer, a second buffer, and a first disk file, wherein the first disk file contains ordered representations of data set addresses;

(b) selecting as a current buffer one of the first and second buffers;

(c) downloading at least one data set that includes addresses of one or more referred data sets;

(d) identifying the addresses of the one or more referred data sets; and

(e) for each identified address:

(e1) generating a representation of the identified address; and

(e2) determining whether the representation is stored in the current buffer without determining whether the representation is stored in the first disk file, and when this determination is negative, storing the representation in the current buffer; and

(f) when the current buffer reaches a predefined full condition:

(f1) selecting the other buffer as the current buffer, wherein the previously current buffer is identified as a non-current buffer;

(f2) ordering representations stored in the non-current buffer; and

(f3) performing an ordered merge of the contents of the non-current buffer into the contents of the first disk file wherein the ordered merge comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

14. (Original) The method of claim 13, further comprising:

in step (e2), when the determination is negative, storing the identified address in the current buffer.

15. (Original) The method of claim 13, further comprising:

in step (e2), when the determination is negative, storing the identified address in a second disk file;

in step (e2), additionally storing with each representation in the current buffer a pointer to the corresponding address stored in the second disk file; and

in step (f2), while ordering the contents of the non-current buffer, keeping with each representation in the non-current buffer its pointer to the corresponding address in the second disk file.

16. (Original) The method of claim 15 wherein

step (e2) comprises: for each representation in the buffer storing an associated flag, setting the flag to a first value when the representation is equal to a representation previously stored in the first disk file, and setting the flag to a second value, distinct from the first value, when the representation is not equal to any representation previously stored in the first disk file; and

step (f) includes: (f4) for each representation whose flag is set to the second value, scheduling the corresponding data set for downloading.

17. (Original) The method of claim 13 wherein step (e2) further comprises:

when a representation in the current buffer is not found in the first disk file during merging, scheduling the corresponding data set for downloading.

18. (Original) The method of claim 13 wherein:

step (a), storing representations of data set addresses, includes storing representations of data set addresses in a sparse disk file which is divided into portions, each portion having a starting address and contents comprising an ordered list of representations of data addresses; and

step (e2), performing an ordered merge of the contents of the current buffer into the contents of the sparse disk file, comprises the following steps:

for each of a plurality of the representations stored in the current buffer:

(e2-1) obtaining a starting address for a corresponding portion of the sparse disk file; and

(e2-2) performing an ordered merge of a subset of the current buffer, starting at the representation for which the starting address was obtained, into the contents of the corresponding portion.

19. (Original) The method of claim 13 wherein:

step (a), storing representations of data set addresses, includes the step of storing representations of data set addresses in a sparse disk file having empty entries interspersed among entries storing said representations; and

step (e2), merging the contents of the buffer with the ordered contents of the sparse disk file, includes:

for each respective representation stored in the buffer:

(e2-1) determining a starting address for a corresponding portion of the sparse disk file; and

(e2-2) sequentially scanning the disk file, starting at the representation for which the starting address was obtained, until the first of (A) a representation matching the respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

20. (Previously Presented) The method of claim 13 wherein the representation of the identified address comprises a checksum of at least a portion of the identified address.

21. (Original) The method of claim 13 wherein step (e2) further comprises:

(e2-1) determining whether the representation is stored in a cache before determining whether the representation is stored in the current buffer;

(e2-2) when the representation is not stored in the cache, and the cache has not reached a predefined full condition, adding the representation to the cache; and

(e2-3) when the representation is not stored in the cache, and the cache has reached said predefined full condition, evicting a stored representation from the cache in accordance with an eviction policy and adding the representation to the cache.

22. (Previously Presented) A method of downloading data sets from among a plurality of host computers, comprising the steps of:

(a) storing representations of data set addresses in a set of data structures, including a buffer and a disk file, wherein representations of data set addresses stored in the disk file are ordered;

(b) downloading at least one data set that includes an address of a referred data set;

(c) identifying the address of the referred data set;

(d) generating a representation of the identified address;

(e) determining whether the representation is stored in the buffer, and whether the disk file is empty;

(f) when the representation is not stored in the buffer and the disk file is empty, scheduling the corresponding data set for downloading;

(g) when the representation is not stored in the buffer and the disk file is not empty, storing the representation in the buffer and delaying scheduling of the corresponding data set for downloading until a condition occurs; and

(h) when it is determined that the condition has occurred, performing an ordered merge of contents of the buffer into contents of the first disk file wherein the ordered merge comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

23. (Previously Presented) A computer program product for use in conjunction with a computer system, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, the computer program mechanism comprising:

a first disk file and a buffer, for storing representations of data set addresses;

a main web crawler module for downloading and processing data sets stored on a plurality of host computers, the main web crawler module identifying addresses of one or more referred data sets in the downloaded data sets; and

an address filtering module for processing a specified one of the identified addresses;

the address filtering module including instructions for:

generating a representation of the identified address;

determining whether the representation is stored in the buffer without determining whether the representation is stored in the first disk file, and when this determination is negative storing the representation in the buffer; and

determining whether the buffer has reached a predefined full condition, and when this determination is positive, ordering the contents of the buffer and then performing an ordered merge of contents of the buffer into the contents of the first disk file wherein the ordered merge comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

24. (Original) The computer program product of claim 23, wherein the address filtering module further includes instructions for storing the identified address in the buffer after determining that the representation is not stored in the buffer.

25. (Original) The computer program product of claim 23, wherein the address filtering module further includes instructions for:

storing the identified address in a second disk file after detenI1ining that the representation is not stored in the buffer; and

storing with each representation in the buffer a pointer to the corresponding address stored in the second disk file; and

during the ordering of the contents of the buffer, keeping with each representation in the buffer its pointer to the corresponding address in the second disk file.

26. (Original) The computer program product of claim 23, wherein

the first disk file is a sparse disk file divided into portions, each portion having a starting address and contents comprising an ordered list of representations of data addresses; and

the address filtering module includes instructions for performing the ordered merge of the ordered contents of the buffer with the contents of the sparse disk file by obtaining a starting address for a sub-file of the sparse disk file, the portion corresponding to one of the representations in the buffer, and performing an ordered merge of a subset of the representations in the buffer, starting at the one representation, into the contents of the portion.

27. (Original) The computer program product of claim 23, wherein

the first disk file is a sparse disk file having empty entries interspersed among entries storing said representations of data addresses; and

the address filtering module includes instructions for performing the ordered merge of the ordered contents of the buffer with the contents of the sparse disk file by obtaining a starting address corresponding to each respective representations in the buffer, and sequentially scanning the first disk file, starting at the starting address, until the first of (A) a representation matching the respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

28. (Original) The computer program product of claim 23 wherein the representation of the identified address comprises a checksum of at least a portion of the identified address.

29. (Original) The computer program product of claim 23, wherein the address filtering module further includes instructions for first determining whether the representation is stored in a cache, and when the first determination is positive, skipping the determination of whether the representation is stored in the buffer.

30. (Original) The computer program product of claim 23, wherein the address filtering module further includes instructions for:

determining whether the first disk file is empty and whether the representation is stored in the buffer; and

if the first disk file is empty and the representation is not stored in the buffer, storing the representation in the buffer and scheduling the corresponding data set for downloading.

31. (Previously Presented) A computer program product for use in conjunction with a computer system, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, the computer program mechanism comprising:

a first disk file, a first buffer, and a second buffer, for storing representations of data set addresses;

a main web crawler module for downloading and processing data sets stored on a plurality of host computers, the main web crawler module identifying addresses of the one or more referred data sets in the downloaded data sets; and

an address filtering module for processing a specified one of the identified addresses; the address filtering module including instructions for:

identifying one of the first and second buffers as a current buffer;

generating a representation of the identified address;

determining whether the representation is stored in the current buffer without determining whether the representation is stored in the first disk file, and when this determination is negative, storing the representation in the current buffer; and

determining whether the current buffer has reached a predefined full condition, and when this determination is positive, selecting the other buffer as the current buffer, wherein the previously current buffer is identified as a non-current buffer, ordering the contents of the non-current buffer and then performing an ordered merge of the contents of the non-current buffer into the contents of the first disk file wherein the ordered merge comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

32. (Original) The computer program product of claim 31, wherein the address filtering module further includes instructions for storing the identified address in the current buffer after determining that the representation is not stored in the current buffer.

33. (Original) The computer program product of claim 31, wherein the address filtering module further includes instructions for:

storing the identified address in a second disk file after determining that the representation is not stored in the current buffer;

storing with each representation in the current buffer a pointer to the corresponding address stored in the second disk file; and

during the ordering of the contents of the non-current buffer, keeping with each representation in the non-current buffer its pointer to the corresponding address in the second disk file.

34. (Original) The computer program product of claim 31, wherein

the first disk file is a sparse disk file divided into sub-files, each sub-file having a starting address and contents comprising an ordered list of representations of data addresses; and

the instructions for performing the ordered merge including instructions for obtaining a starting address for a sub-file of the first disk file, the sub-file corresponding to one of the representations in the buffer, and performing an ordered merge of a subset of the representations in the non-current buffer, starting at the one representation, into the contents of the sub-file.

35. (Original) The computer program product of claim 31, wherein

the first disk file is a sparse disk file having empty entries interspersed among entries storing said representations of data addresses; and

the address filtering module includes instructions for performing the ordered merge of the ordered contents of the buffer with the contents of the sparse disk file by obtaining a starting address corresponding to each respective representations in the buffer, and sequentially scanning the first disk file, starting at the starting address, until the first of (A) a representation matching the respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

36. (Original) The computer program product of claim 31 wherein the representation of the identified address comprises a checksum of at least a portion of the identified address.

37. (Original) The computer program product of claim 31, wherein the address filtering module further includes instructions for:

determining whether the first disk file is empty and whether the representation is stored in the current buffer; and

if the first disk file is empty and the representation is not stored in the current buffer, storing the representation in the current buffer and scheduling the corresponding data set for downloading.

38. (Previously Presented) A web crawler for downloading data set addresses from among a plurality of host computers, comprising:

a first disk file and a buffer, for storing representations of data set addresses;

a main web crawler module for downloading and processing data sets stored on a plurality of host computers, the main web crawler module identifying addresses of the one or more referred data sets in the downloaded data sets; and

an address filtering module for processing a specified one of the identified addresses; the address filtering module including instructions for:

generating a representation of the identified address;

determining whether the representation is stored in the buffer without determining whether the representation is stored in the first disk file, and when this determination is negative storing the representation in the buffer; and

determining whether the buffer has reached a predefined full condition, and when this determination is positive, ordering the contents of the buffer and then performing an ordered merge of the contents of the buffer into the contents of the first disk file wherein the ordered merge comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

39. (Original) The web crawler of claim 38, wherein the address filtering module further includes instructions for storing the identified address in the buffer following a determination that the representation is not stored in the buffer.

40. (Original) The web crawler of claim 38, wherein the address filtering module further includes instructions for:

storing the identified address in a second disk file after determining that the representation is not stored in the buffer; and

storing with each representation in the buffer a pointer to the corresponding address stored in the second disk file; and

during the ordering of the contents of the buffer, keeping with each representation in the buffer its pointer to the corresponding address in the second disk file.

41. (Original) The web crawler of claim 38 wherein

the first disk file is a sparse disk file divided into portions, each portion having a starting address and contents comprising an ordered list of representations of data addresses; and

the address filtering module further includes instructions for:

obtaining, from an index, a starting address for a portion in the sparse disk file corresponding to one of the representations stored in the buffer; and

performing an ordered merge of a subset of the representations stored in the buffer, starting at the representation for which the starting address was obtained, into the contents of the corresponding portion.

42. (Original) The web crawler of claim 38 wherein

the first disk file is a sparse disk file having empty entries interspersed among entries storing said representations of data addresses; and

the address filtering module includes instructions for performing the ordered merge of the ordered contents of the buffer with the contents of the sparse disk file by obtaining a starting address corresponding to each respective representations in the buffer, and sequentially scanning the first disk file, starting at the starting address, until the first of (A) a representation matching the respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

43. (Original) The web crawler of claim 38 wherein the representation of the identified address comprises a checksum of at least a portion of the identified address.

44. (Original) The web crawler of claim 38 wherein the address filtering module further includes instructions for:

determining whether the representation is stored in a cache before determining whether the representation is stored in the buffer, and when this determination is negative, determining whether the representation is stored in the buffer;

when the second determination is negative, storing the representation in the buffer;

when the first determination is negative, and predefined other criteria are met, storing the representation in the cache; and

when the cache has reached a predefined full condition, evicting a stored representation from the cache in accordance with an eviction policy.

45. (Original) The web crawler of claim 38 wherein the address filtering module further includes instructions for detem1ining whether the first disk file is empty and whether the representation is stored in the buffer, and if the first disk file is empty and the representation is not stored in the buffer, storing the representation in the buffer and scheduling the corresponding data set for downloading.

46. (Previously Presented) A web crawler for downloading data set addresses from among a plurality of host computers, comprising:

a first disk file, a first buffer and a second buffer, for storing representations of data set addresses;

a main web crawler module for downloading and processing data sets stored on a plurality of host computers, the main web crawler module identifying addresses of the one or more referred data sets in the downloaded data sets; and

an address filtering module for processing a specified one of the identified addresses; the address filtering module including instructions for:

identifying one of the first and second buffers as a current buffer;

generating a representation of the identified address;

determining whether the representation is stored in the current buffer without determining whether the representation is stored in the first disk file, and when this determination is negative, storing the representation in the current buffer; and

determining whether the current buffer has reached a predefined full condition, and when this determination is positive, selecting the other buffer as the current buffer, wherein the previously current buffer is identified as a non-current buffer, ordering the contents of the non-current buffer and then performing an ordered merge of the contents of the non-current buffer into the contents of the first disk file wherein the ordered merge

comprises preventing duplication of any of the representations of data set addresses stored in the first disk file.

47. (Original) The web crawler of claim 46, wherein the address filtering module further includes instructions for storing the identified address in the current buffer after determining that the representation is not stored in the current buffer.

48. (Original) The web crawler of claim 46, wherein the address filtering module further includes instructions for:

storing the identified address in a second disk file after determining that the representation is not stored in the current buffer;

storing with each representation in the current buffer a pointer to the corresponding address stored in the second disk file; and

during the ordering of the contents of the non-current buffer, keeping with each representation in the non-current buffer its pointer to the corresponding address in the second disk file.

49. (Original) The web crawler of claim 46, wherein

the first disk file is a sparse disk file divided into sub-files, each sub-file having a starting address and contents comprising an ordered list of representations of data addresses; and

the instructions for performing the ordered merge including instructions for obtaining a starting address for a sub-file of the first disk file, the sub-file corresponding to one of the representations in the buffer, and performing an ordered merge of a subset of the

representations in the non-current buffer, starting at the one representation, into the contents of the sub-file.

50. (Original) The web crawler of claim 46 wherein

the first disk file is a sparse disk file having empty entries interspersed among entries storing said representations of data addresses; and

the address filtering module includes instructions for performing the ordered merge of the ordered contents of the buffer with the contents of the sparse disk file by obtaining a starting address corresponding to each respective representations in the buffer, and sequentially scanning the first disk file, starting at the starting address, until the first of (A) a representation matching the respective representation is found and (B) one of the empty entries is found, and when an empty entry is found storing the respective representation in the empty entry.

51. (Original) The web crawler of claim 46 wherein the representation of the identified address comprises a checksum of at least a portion of the identified address.

52. (Original) The web crawler of claim 46, wherein the address filtering module further includes

instructions for: determining whether the first disk file is empty and whether the representation is stored in the current buffer; and

when the first disk file is empty and the representation is not stored in the current buffer, storing the representation in the current buffer and scheduling the corresponding data set for downloading.